Conversational Hate-Offensive detection in Code-Mixed Hindi-English Tweets

Ratnavel Rajalakshmi[1], S Srivarshan[1], Faerie Mattins[1], E Kaarthik[1], Prithvi Seshadri[1] and Anand Kumar M.[2]

[1]*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai*

[2]*Department of Information Technology, National Institute of Technology Karnataka (NITK), Surathkal*

Abstract

Hate speech in social media has increased due to the increased use of online forums for sharing the opinion among the people. Especially, people prefer expressing the views in their native language while posting such objectionable contents in many social media platforms. It is a challenging task to have an automated system to identify such hate and offensive tweets in many regional languages due to the rich linguistics nature. Recently, this problem has become too complicated, due to the use of multi-lingual and code-mixed tweets. The code-mixed data includes the mixing of two languages on the granular level. A word that might not be a part of either language may be found in the data. To address the above challenges in Hindi-English tweets, we propose an efficient method by combining the IndicBERT with an effective ensemble based method. We have applied different methodologies to find a way to accurately classify whether the given tweet is considered to be Hate Speech or Not in code-mixed Hinglish dataset. Three different models namely, IndicBERT, XLM Roberta and Masked LM were used to embed the tweet data. Then various classification methods such as Logistic Regression, Support Vector Machine, Ensembling and Neural Networks based method were applied to perform classification. From extensive experiments on the data set, embedding the code-mixed data with IndicBERT and Ensembling was found to be the best method, which resulted in an macro F1-score of 62.53%. This work was submitted to the shared task of the HASOC 2021 [1] [2] Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages Competition by team TNLP.

Keywords

Code-mixed tweets, Natural Language Processing, Sentiment Analysis, Machine Learning,

1. Introduction

A well-known proverb aptly describes India's linguistic diversity: "Chaar kos par baani, kos-kos par badle paani"(The language spoken in India changes every few kilometres, just like the taste of the water). Our country contains 30 languages, each of which is spoken by over a million people. These 30 languages are merely a linguistic window through which we can observe the 122 languages spoken by at least 10,000 people each. Then there are the 1,599 languages, the majority of which are dialects limited to certain geographical areas. Although there is a diverse collection of languages in India, only Hindi has adequate research in terms of NLP. Presently, much effort is placed towards the creation of multi-lingual instead of many models

Forum for Information Retrieval Evaluation, December 13-17, 2021, India

rajalakshmi.r@vit.ac.in (R. Rajalakshmi)
0000-0002-6570-483X (R. Rajalakshmi)

for individual language. However, choosing a language to convert poses a problem as it should be understood and easily interpreted by the end-user.

Although English is not the language most people speak globally, it is globally acknowledged and used in essential parts of government and corporations. Given that the majority of NLP research is conducted in English, it has proven to be the ideal language for conversion. When it comes to Indian languages, the languages are morphologically rich, resulting in greater complexity in literature and sentence formation. It results in a deficit of datasets, as the meaning cannot be analyzed until a large amount of data, which are structurally similar, is used. For example, “Mein phal khaatha hun” and “Mein khaatha hun phal”. Unfortunately, the progress of NLP is stunted by the lack of availability of evaluation benchmarks and large-scale datasets specific for Indian languages. Therefore, models try to work with a limited data set and get the maximum accuracy possible.

In this paper, the related works have been analysed and discussed in section 2. An architecture diagram consisting of the models acts as an overview of what has been used, illustrated in section 3. Detailed information about the data set and pre-processing is given in subsection 3. Three different embedding models were used for embedding, which is explained in the 3rd subsection, and also the classifier models used are explained in the same section. The results are discussed in section 4, following with the conclusion of the paper. This work has been submitted to the shared task of the HASOC 2021 [1] [2] Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages Competition by team TNLP.

2. Related Works

Gaurav Arora and Jio Haptik [3] used iNLTK in place of Semantic Evaluation. It provided insights on Data Augmentation, Textual Similarity, Sentence Embeddings, Word Embeddings, Tokenization and Text Generation in 13 Indic Languages. Xiaozhi Ou et al. [4] Used XLM-RoBERTa and observed a 4.86% increment in accuracy compared to XLM. Bertelt Braaksma et al. [5] use LSTM and HuggingFace Transformers and show us that variation in negative classes plays an important role. Alex Wang et al. [6] use General Language Understanding Evaluation (GLUE). It confirms the utility of attention mechanisms and transfers learning methods such as ELMo in NLU systems, which combine to outperform the best sentence representation models on the GLUE benchmark. Alexis Conneau et al. [7] Make use of XNLI, a model derived from Multi-Genre Natural Language Inference Corpus (MultiNLI), to evaluate Cross-lingual Sentence Representations. Several approaches are based on cross-lingual sentence encoders and machine translation systems. Ratnavel Rajalakshmi, Yashwanth Reddy [8] performed HASOC detection for Multilingual tweets. The model is built by translating the Tweets, after which feature extraction and data imbalance is handled. Adina Williams et al. [9] make use of MultiNLI, a dataset used for the development and evaluation of machine learning models for sentence understanding. Thara S et al. [10] performed comparative analysis of different code-mixed natural language processing (NLP) approaches based on their accuracy. Although the implementation consisted of various real-time applications such as e Parsing, Machine

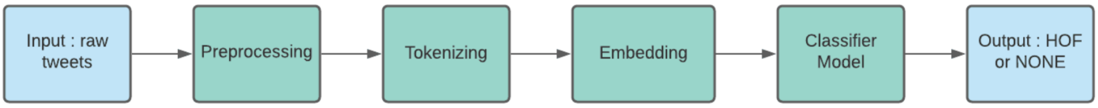

Figure 1: Architecture of the proposed methodology

Translation (MT), Automatic Speech Recognition (ASR), Information Retrieval (IR) and many more, the advantages were clearly described. Atharva Kulkarni et al. [11] Used deep learning models to classify Marathi text. Out of all the models used, IndicBERT and BiLSTM had the highest accuracy, just above 94%. Divyanshu Kakwani et al. [12] Introduced pre-trained multilingual language models for Indian languages and evaluated using various Indic models such as IndicCorp, IndicGLUE, IndicFT and IndicBERT. Ratnavel Rajalakshmi et al. [13] used LSTM, BLSTM and BERT based models to perform comparative analysis on Transformer based approach in Code-Mixed Tamil for Offensive Language. The paper also introduced vector representation such as TD-IDF and Glove. The authors [14] performed feature extraction with Ensemble Methods for Hate speech identification. Since German and Hindi were used, specific methods such as SMOTE Analysis, Mutual Information and Chi-Square were implemented. In another work, [15] they processed Hinglish (Mix of Hindi and English) tweets and trained a machine learning model for sentiment analysis. The model used Boosting along with classifiers to improve the efficiency of the algorithm. Applying machine learning models and deep learning algorithms are common across various applications including web page categorization [16, 17, 18] and sign detection [19]. Sentiment analysis in English is a most common problem and many approaches are suggested by various researchers by applying both machine learning and deep learning techniques [20, 21, 22]. To address the challenges in code-mixed Hindi-English tweets, different relevance factor has been proposed in [23], to find the borrowing likeliness in bi-lingual tweets.

3. Proposed Methodology

The proposed work is a system where the input text is raw tweets consisting of offensive and non-offensive tweets. These tweets are preprocessed and cleaned for easier processing of the data. Next, this cleaned data is sent to the tokenizer, where the data is tokenized. The tokenizer also does padding and truncation of data. Following this process is embedding. These tokenized data are embedded using the appropriate embedding models such as IndicBERT, XLMRoberta, and MaskedLM. Further, these embedded outputs are saved and sent to the classifier model. These classifiers are trained with the embedded data along with the corresponding outputs. Finally, the predicted output is given, predicting whether the tweet is hate and offensive (HOF) tweets or non-hate offensive (NONE) tweets. Figure 1 illustrates the architecture of the proposed model.

3.1. Dataset

We have considered the dataset released as part of the HASOC-2021 competition's Subtask 2 : Identification of Conversational Hate-Speech in Code-Mixed Languages (ICHCL) dataset. The dataset consists of Hindi, English and code-mixed Hindi tweets. The task is to classify these tweets as non-hate offensive (NONE) or hate and offensive (HOF) tweets (HASOC 2021). The hate and offensive tweets consist of the tweets which represent offensive, hate speech and profane content. The non-hate offensive tweet consists of regular tweets. A conversational thread also can comprise hate and offensive content material, which is not always obvious simply from the single comment or the response to the comment. However, it may be recognized in the context of the discern content material is given. Hence, the dataset has a tree structure consisting of the source tweet, reply tweet, and comments. There are a total of 5740 tweets in the training dataset, which are classified into 2841 HOF tweets and 2899 NONE tweets. There are 1348 tweets in the testing dataset, which has a combination of HOF and NONE tweets.

3.2. Preprocessing

Each entry in the dataset is made up of tweets and their corresponding comments in a tree-like structure. These tweets and comments were flattened into a single line by appending all the child nodes to the original tweet. Each tweet will have a tweet ID and its label. These flattened texts were then compiled into a new DataFrame which is now considered the data to train the model. The tweets consist of phrases and words that may complicate the model like mentions (e.g., @StevieG), URLs (e.g., www.youtube.com), emojis, unwanted newline, retweet tags(e.g., RT), and hashtags (e.g., #sunnyday). The tweets are then processed one by one, and any phrases similar to the ones mentioned above are removed from the tweets. Stop words are the most commonly used words in that language that are generally removed because they do not hold any information concerning the task at hand. Similarly, the tweets also have multiple stopwords (e.g., above, below, is, that, etc.), which must be removed in the preprocessing step. Hence, for English stopwords removal, the NLTK library is used, and for Hindi, a list of predefined stop words used in Hindi is saved into a text file and are removed from the tweets in the preprocessing step. Stemming is the process by which some derived words are converted to their root forms. (e.g., argued, arguing are converted to argue). Both the Hindi and English texts contain words that can be stemmed. For English words, SnowballStemmer is used to perform the steaming process, and for Hindi words, a list of Hindi stemmers used was produced by students of Banasthali University [24], this is used here too to perform stemming. Thus the dataset is subjected to an algorithm that goes through each word and stems it. Figure 2 illustrates the preprocessing carried out for the raw tweets. Table 1 and table 2 give some examples of the dataset before and after preprocessing respectively.

3.3. Embedding

IndicBERT [12] is a multilingual model derived from the ALBERT model for Natural Language Processing tasks. The authors have supposedly trained it on a novel corpus of more than 9 billion tokens. It has also been tested on diverse NLP tasks. It is better than the other models as it uses around a tenth of the parameters, enabling it to process data much faster than other

Table 1
Condensed dataset before preprocessing

tweet_id	text	label
1392450559693651968	yaar nahi hoti padhai leaven knows what's wrong...	HOF
1393625939741794306	@IshanKa75853640 @deashingmilan Abe kapoor khandan...	NONE
1394176816055660000	And Modi #bhakts are busy drinking Gaumutra and...	NONE
1392421887578312711	Bhakton mooh chupao sharm se Baklol sab #IndiaWithPalestine	HOF
1396169865979863041	@kunalkamra88 Doctors aur Scientists se manga hai...	HOF

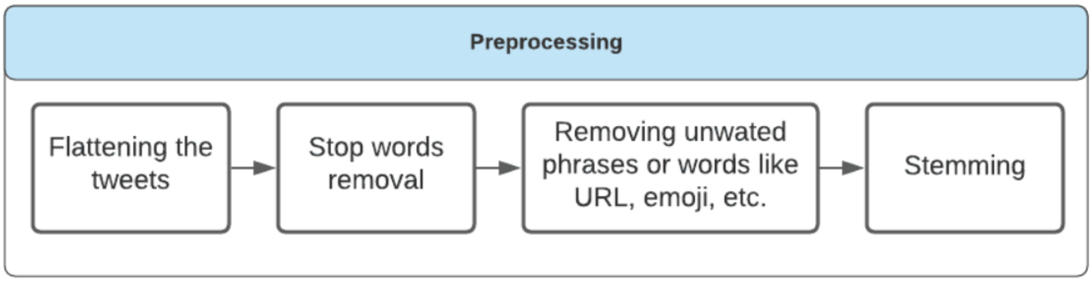

Figure 2: Preprocessing the data

Table 2
Condensed dataset after preprocessing

tweet_id	text	label
1392450559693651968	yaar nahi hoti padhai leaven know what wrong...	HOF
1393625939741794306	abe kapoor khandan...	HOF
1394164896904802306	and modi #bhakt busi drink gaumutra...	NONE
1392421887578312711	bhakton mooh chupao sharm se baklol sab #indiawithpalestin	HOF
1396169865979863041	doctor aur scientist se manga hai...	HOF

models. However, this fact does not compromise its performance as it achieves results almost as good as or sometimes even better than comparable models. The Albert model from which IndicBERT is derived is in itself a derivative of the BERT model. This model covers 12 main Indian languages such as English, Bengali, Assamese, Hindi, Gujarati, Malayalam, Kannada, Oriya, Marathi, Telugu, Tamil and Punjabi. The monolingual data on which the model is trained is available on the author's website.

The model is made available for public use with the help of the Transformers [25] Library courtesy of the HuggingFace [26] website. The Transformers Python Module provides a helpful pipeline to download and load the IndicBERT Tokenizer and the Model. Now the preprocessed dataset is subjected to the Tokenizer. The entire dataset is taken as bulk and given as the input for the Tokenizer. However, due to memory constraints on the host machine, the entire tokenized dataset cannot be fed to the model. Therefore we had to split the tokenized dataset. However, the tokenized input exists as a Pytorch Tensor which cannot be split. Thus the output of the Tokenizer is converted into a Numpy array. Now the corresponding token for each dataset is extracted from the array. These extracted tokens are then again converted back to a Pytorch Tensor format. Now these individual tokens can be passed to the model to get the embedded output. Due to the memory constraints, the tokenized input is split into equal parts, consisting of 200 entries. These parts are then fed to the Model one by one, and the embedded outputs are saved to the disk for further processing. To save and load the embeddings, the Pickle Python Module has been used.

XLM-RoBERTa is a multilingual RoBERTa version. It has been pre-trained on 2.5 TB of filtered CommonCrawl data with 100 languages. RoBERTa is a transformer model which is self-supervised and trained with a vast corpus of data. Self-supervised means that this model was pre-trained exclusively on raw texts which wasn't labelled by an human and consists of an automatic process to produce inputs and its respective labels from those texts. It makes the model more efficient since this can be utilized in many scenarios and datasets. XLM-RoBERTa was pre-trained using the Masked language modelling (MLM) objective. Similar to the IndicBERT model, the XLM-RoBERTa model is made available for public use with the help of the Transformers [25] Library courtesy of the HuggingFace [26] website. Transformers are used to download and load the XLM-RoBERTa tokenizer and model. Again, due to the memory constraints, the tokenized data is split into smaller chunks and fed in the model similar to the IndicBERT model. These tokens are passed to the model to get the embedded output and are saved to the disk for further processing.

Prior to the addition of BERT sequences, 15% of the words in each training sample is replaces with a MASK token. The model is then trained to identify the original word that was masked based on the context of the rest of the training sample. Most MaskedLM models utilise the following methods to accomplish said tasks- Addition of a classification layer along with encoder, Conversion of product of vector outputs and embedding matrix into vocabulary dimension and Probability of each word. However, the major disadvantage of the BERT loss function is that only the prediciton of masked values is considered, ignoring the non masked words. In our model, the tokenizer uses BertTokenizer and consists of a pre-trained model called 'bert-base-uncased' and the model uses BertForMaskedLM which also makes use of the same pre-trained model. The tokenized input is padded and truncated with return_tensors as pre-trained PyTorch tensor.

3.4. Classifiers

3.4.1. Logistic Regression

Logistic Regression is a straightforward machine learning model. It is derived from the Linear Regression Model. Binary Regression is when the output from the Linear Regression model is subjected to a cost function that constrains the output into binary values, that is, 0 or 1. This cost function uses the principle of a threshold. That is, if a value is above the given threshold, then the output is 1. If the value is lower than the threshold, then the value is 0. This way, the Simple Linear Regression model is used for binary classification, also known as Binary Logistic Regression. In this work, we load the embedded outputs from the disk. The corresponding labels for the data are also loaded. It is the training data for the Logistic Regression model. Now, this data is split into training data and validation data for validating the model. The Linear Regression Model is loaded using the Scikit-Learn [27] Library for Python. Two parameters were modified in the Logistic Regression model. The maximum number of iterations was set to 1000 and the random state was set to 11. The rest of the parameters are the same as the default values. The model is then trained on the training data. Now the trained model is validated, and the results are recorded.

3.4.2. Support Vector Machines (SVM)

Support vector machines (SVMs) are supervised learning algorithms that can be used for outlier detection,classification and regression. SVM is successful in high-dimensional domains and uses a portion of training points, commonly known as support vectors, in the decision function, which makes it memory economical. The goal of the this algorithm is to discover a hyperplane that distinguishes between data points in an N-dimensional space. Here, N indicates the number of features. There are a variety of hyperplanes from which to divide the two types of data points. The primary objective is to choose a plane which has a significant distance between data points from both classes or the one which has the largest margin. By Increasing the margin distance, there is some reinforcement, making subsequent data points easier to classify. In this work, we load the embedded outputs from the disk and the corresponding labels for the data. It is the training data for the SVM model. Now, this data is split into training data and validation data for validating the model. The SVM Model is loaded using the Scikit-Learn [27] Library from Python. Three parameters were modified in the SVM model. The maximum number of iterations was set to 100, random state was set to 11 and RBF was chosen as the kernel. The model is then trained on the training data. Now the trained model is validated, and the results are recorded.

3.4.3. Ensemble - Majority Voting

Ensembling is a machine learning method in which multiple weak learners/models are trained on a given data and combined to give each model better accuracy. In this work, we have made use of one such Ensembling technique - Majority Voting. Majority Voting in its purest essence is a technique in which a model makes a prediction that was predicted majorly by multiple other weak models. In a classification task, the predictions of the weak models are considered to be votes. These votes are tallied, and the prediction with the majority of the votes is the final prediction of that particular set of input variables. Various models make up the weak learners that contribute to the voting. The learners that we have used in this task are Logistic Regression, Stochastic Gradient Descent, Naive Bayes, Random Forest and Decision Tree. First, the embedded outputs are loaded from the disk. The corresponding labels are also loaded. The data is split into a training set and validation set. The multiple classifiers are trained on the training set one by one. After training is complete, the testing data is passed on to the models.

The predictions of all the models are considered as votes, and majority voting takes place. The final prediction is obtained at this step.

3.4.4. Neural Network

Neural Networks are algorithms which try to emulate the human brain's behaviour and allow the computer to solve complex problems that it usually cannot. They generally comprise multiple layers of nodes. These nodes can be broadly classified as input nodes, hidden nodes and output nodes. Any Artificial Neural Network Model with more than one hidden layer is generally classified as a Deep Learning Model. Each node is connected to another node. Each node has some properties associated with it, such as a corresponding weight and a threshold value. Neural Networks are trained on data. They become more accurate as they are trained on more data and for a longer time. The Artificial Neural Network used in this model consists of a single input layer of size 32 and an activation layer with the ReLU function. The hidden layer consists of 4 layers. Two Dense layers sandwiched between two dropout layers consist of the hidden layers. The model tended to over-fit the data, which led to the implementation of the dropout layers. Finally, we have the output layer which provides a singular output and the activation function associated with it is the Sigmoid function. Then the model, with its loss function set to Binary Cross Entropy and the optimizer set to Stochastic Gradient Descent, was compiled. The model was run for 1000 epochs, and early stopping was employed. In this work, we load the embedded outputs from the disk. The corresponding labels for the data are also loaded. Now, this dataset is divided into training dataset and validation dataset for validating the model. The network is then trained on the training dataset. Now the trained network is validated, and the results are recorded.

4. Results and Discussion

The lesser number of parameters in the IndicBERT model was one of the main reasons as to which it was specifically chosen as the base model. This feature makes it easier to train the model and get predictions. Further, this model has already been pre-trained in more than 12 Indian languages. This was done to utilize the relatedness among the pre-trained data and the code-mixed data used for training in this work. Although the same is possible in MaskedLM, it has not been pre-trained in Indian languages, and the resources spent to get the same performance are significantly higher. This is the reason why MaskedLM has a lower macro F1-score of 50.34% than IndicBERT and XLMRoberta. Due to this reason, a submission was not made for MaskedLM. XLMRoberta shows low macro F1-score of 51.43% compared to the other models. One of the reasons why IndicBERT performed better than XLMRoberta is because the data in IndicBERT was comparatively smaller than the data produced by XLMRoberta after embedding the data. It led to better performance while training IndicBERT; as for XLMRoberta, it was necessary to use PCA to reduce the number of data features. Hence, IndicBERT gave a better macro F1-score and results compared to XLMRoberta. In this research, two classifiers were applied on IndicBERT, Ensembling using majority voting and ANN. It is noticed that Ensembling gave a slightly better results than the results obtained using the ANN classifier. The evaluation metrics with the embedding type and its classifiers can be viewed in Table 1.

Table 3
Quantitative performance validation results of different types of embedding model used with their classifiers (* denotes macro averaged values)

Submission Name	Types of Embedding	Classifier	Precision*	Recall*	F1-Score*
-	Masked LM	Logistic Regression	50.35%	50.35%	50.34%
tNLP_CMH_S2_X	XLM-RoBERTa	SVM	53.19%	52.77%	51.43%
tNL_CMH_S3	IndicBERTt	ANN	61.89%	61.83%	61.82%
tNLP_CMH_S1	IndicBERT	Ensembling	62.63%	62.62%	62.53%

5. Conclusion and Future Work

This work was submitted to the FIRE-2021 shared task on Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages (HASOC 2021). The submission obtained the 13th rank among the other submissions. In this research, the problem of identifying the conversational hate speech in code-mixed languages have been experimentally studied for English and Hindi tweets. The importance of tokenizing and embedding the data using different approaches have been analyzed in this research using IndicBERT, XLMRoberta and MaskedLM. Also, multiple classifiers were used on these embedded models, and the classifiers with the best performances were discussed. IndicBERT gave the best experimental results with the highest macro F1-score of 62.53% compared to the other embedding methods used. The models for embedding the data have been restricted to only the three models mentioned above in this work. However, none of these models was trained on code-mixed data and was trained on languages or text containing single languages. Further models that were trained on code-mixed data can be explored in the future.

References

[1] S. Modha, T. Mandl, G. K. Shahi, H. Madhu, S. Satapara, T. Ranasinghe, M. Zampieri, Overview of the HASOC Subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages and Conversational Hate Speech, in: FIRE 2021: Forum for Information Retrieval Evaluation, Virtual Event, 13th-17th December 2021, ACM, 2021.

[2] S. Satapara, S. Modha, T. Mandl, H. Madhu, P. Majumder, Overview of the HASOC Subtrack at FIRE 2021: Conversational Hate Speech Detection in Code-mixed language , in: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, CEUR, 2021.

[3] G. Arora, inltk: Natural language toolkit for indic languages, CoRR abs/2009.12534 (2020). URL: https://arxiv.org/abs/2009.12534. arXiv:2009.12534.

[4] X. Ou, H. Li, Ynu@dravidian-codemix-fire2020: Xlm-roberta for multi-language sentiment analysis, in: FIRE, 2020.

[5] B. Braaksma, R. Scholtens, S. van Suijlekom, R. Wang, A. Üstün, Fissa at semeval-2020 task 9: Fine-tuned for feelings, CoRR abs/2007.12544 (2020). URL: https://arxiv.org/abs/2007.12544. arXiv:2007.12544.

[6] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. R. Bowman, GLUE: A multi-task benchmark and analysis platform for natural language understanding, CoRR abs/1804.07461 (2018). URL: http://arxiv.org/abs/1804.07461. arXiv:1804.07461.

[7] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, V. Stoyanov, XNLI: evaluating cross-lingual sentence representations, CoRR abs/1809.05053 (2018). URL: http://arxiv.org/abs/1809.05053. arXiv:1809.05053.

[8] B. YashwanthReddy, R. Rajalakshmi, Dlrg@hasoc 2020: A hybrid approach for hate and offensive content identification in multilingual tweets, in: FIRE, 2020.

[9] A. Williams, N. Nangia, S. R. Bowman, A broad-coverage challenge corpus for sentence understanding through inference, CoRR abs/1704.05426 (2017). URL: http://arxiv.org/abs/1704.05426. arXiv:1704.05426.

[10] S. Thara, E. S., B. V., M. VidhyaSaiBhagavan, M. PhanindraReddy, Code mixed question answering challenge using deep learning methods, 2020 5th International Conference on Communication and Electronics Systems (ICCES) (2020) 1331–1337.

[11] A. Kulkarni, M. Mandhane, M. Likhitkar, G. Kshirsagar, J. Jagdale, R. Joshi, Experimental evaluation of deep learning models for marathi text classification, CoRR abs/2101.04899 (2021). URL: https://arxiv.org/abs/2101.04899. arXiv:2101.04899.

[12] D. Kakwani, A. Kunchukuttan, S. Golla, G. N.C., A. Bhattacharyya, M. M. Khapra, P. Kumar, IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 4948–4961. URL: https://aclanthology.org/2020.findings-emnlp.445. doi:10.18653/v1/2020.findings-emnlp.445.

[13] R. Rajalakshmi, Y. Reddy, L. Kumar, DLRG@DravidianLangTech-EACL2021: Transformer based approachfor offensive language identification on code-mixed Tamil, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, Association for Computational Linguistics, Kyiv, 2021, pp. 357–362. URL: https://aclanthology.org/2021.dravidianlangtech-1.53.

[14] R. Rajalakshmi, B. Y. Reddy, Dlrg@hasoc 2019: An enhanced ensemble classifier for hate and offensive content identification, in: FIRE, 2019.

[15] R. Rajalakshmi, P. Reddy, S. Khare, V. Ganganwar, Dlrg@cis 2021: Sentimental analysis of code-mixed hindi language tweets, in: CIS, 2021.

[16] R. Rajalakshmi, Supervised term weighting methods for url classification, Journal of Computer Science 10 (2014) 1969–1976. doi:10.3844/jcssp.2014.1969.1976.

[17] R. R, A. C., An effective and discriminative feature learning for url based web page classification, 2018. doi:10.1109/SMC.2018.00240.

[18] R. Rajalakshmi, H. Tiwari, J. Patel, A. Kumar, R. Karthik., Design of kids-specific url classifier using recurrent convolutional neural network, Procedia Computer Science 167 (2020) 2124–2131. doi:https://doi.org/10.1016/j.procs.2020.03.260.

[19] S. V, A. S, B. R, R. R, Autonomous driving system with road sign recognition using convolutional neural networks, 2019. doi:10.1109/ICCIDS.2019.8862152.

[20] V. Ganganwar, R. Rajalakshmi, Implicit aspect extraction for sentiment analysis: A survey of recent approaches, Procedia Computer Science 165 (2019) 485–491. URL: https://www.sciencedirect.com/science/article/pii/S1877050920300181. doi:https://doi.org/10.

1016/j.procs.2020.01.010.

[21] S. Sivakumar, R. R, Analysis of sentiment on movie reviews using word embedding self-attentive lstm, 2021. doi:10.4018/IJACI.2021040103.

[22] S. Sivakumar, R. Rajalakshmi, Hybrid convolutional bidirectional recurrent neural network based sentiment analysis on movie reviews, Computational Intelligence 37 (2021) 735–757.

[23] R. Rajalakshmi, R. Agrawal, Borrowing likeliness ranking based on relevance factor, in: Proceedings of the Fourth ACM IKDD Conferences on Data Sciences, CODS '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: https://doi.org/10.1145/3041823.3067694. doi:10.1145/3041823.3067694.

[24] S. Paul, N. Joshi, I. Mathur, Development of a hindi lemmatizer, CoRR abs/1305.6211 (2013). URL: http://arxiv.org/abs/1305.6211. arXiv:1305.6211.

[25] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Huggingface's transformers: State-of-the-art natural language processing, CoRR abs/1910.03771 (2019). URL: http://arxiv.org/abs/1910.03771. arXiv:1910.03771.

[26] Hugging face – the ai community building the future., https://huggingface.co/, 2020.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, CoRR abs/1201.0490 (2012). URL: http://arxiv.org/abs/1201.0490. arXiv:1201.0490.